RustNL GOSIM 2024 EUROPE

# MODULAR servo

# Modular Servo:

# What does it mean, where do we stand, and where are we going?

**Gregory Terzian**
Servo TSC
member(maintainer)

May 6th, 2024

Servo, the embeddable, independent, memory-safe, **modular**, parallel web rendering engine

**modularity(Wikipedia):** the degree to which a **system**'s components may be separated and recombined, often with the benefit of:

1. flexibility and variety in use**.**
2. reduction in complexity(hide complexity of component behind interface)

**system(Wikipedia):** a group of interacting or interrelated elements that act according to a set of rules to form a unified whole

Servo, the embeddable, independent, memory-safe, **modular**, parallel web rendering engine

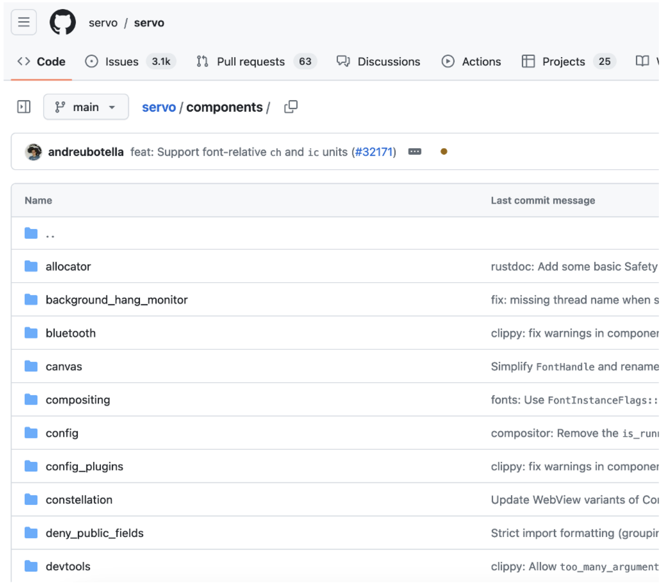
1. Modularity of internal components
2. As a modular component embedded in another system
3. Components of Servo used in other systems(and Servo using other system’s components)

## Modularity of components:

1. Do components hide complexity from each other?
2. Can components be separated and recombined?

Example: Image cache

**Servo as a modular system:** a group of interacting elements, components(layout, script, networking, etc...), forming a web engine.


servo / servo

Code | Issues 3.1k | Pull requests 63 | Discussions | Actions | Projects 25

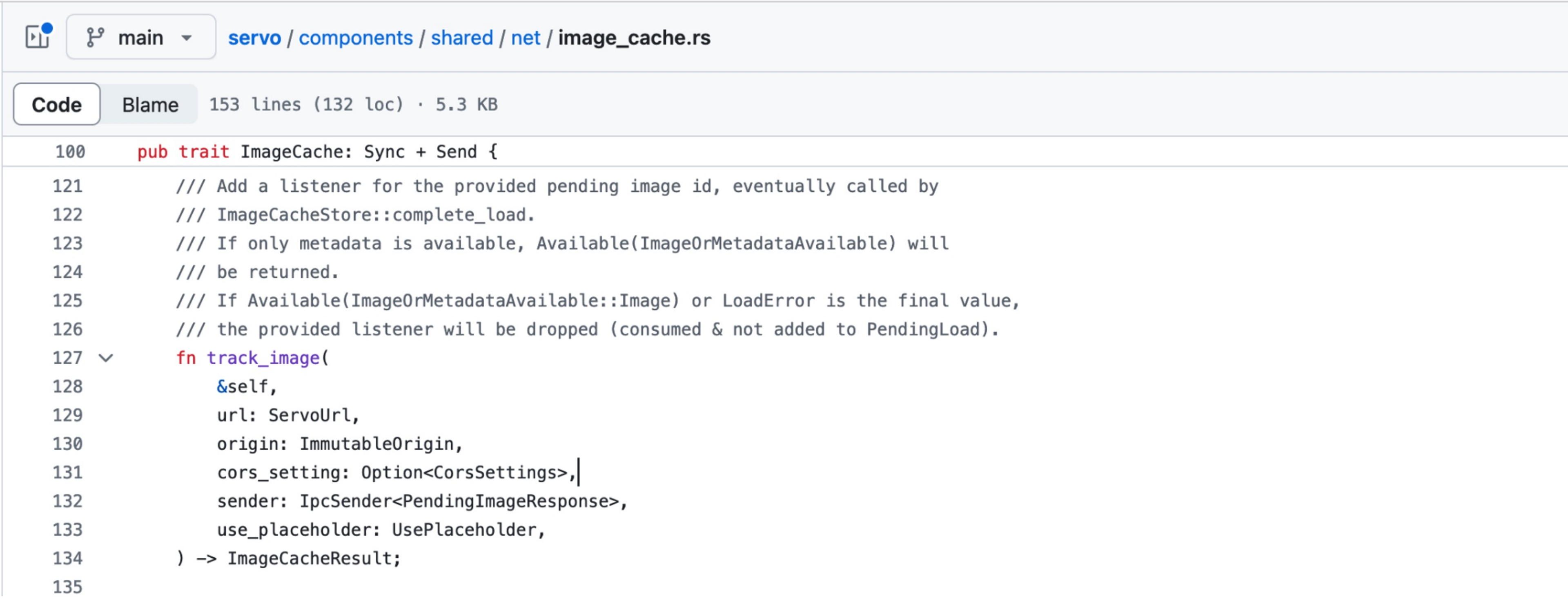
main

servo / components /

andreubotella feat: Support font-relative ch and ic units (#32171)

| Name | Last commit message |
|---|---|
| .. | |
| allocator | rustdoc: Add some basic Safety |
| background_hang_monitor | fix: missing thread name when s |
| bluetooth | clippy: fix warnings in componer |
| canvas | Simplify `FontHandle` and rename |
| compositing | fonts: Use `FontInstanceFlags::` |
| config | compositor: Remove the `is_runı` |
| config_plugins | clippy: fix warnings in componer |
| constellation | Update WebView variants of Coı |
| deny_public_fields | Strict import formatting (groupir |
| devtools | clippy: Allow `too_many_argument` |

Modular?


main

servo / components / shared / net / image_cache.rs

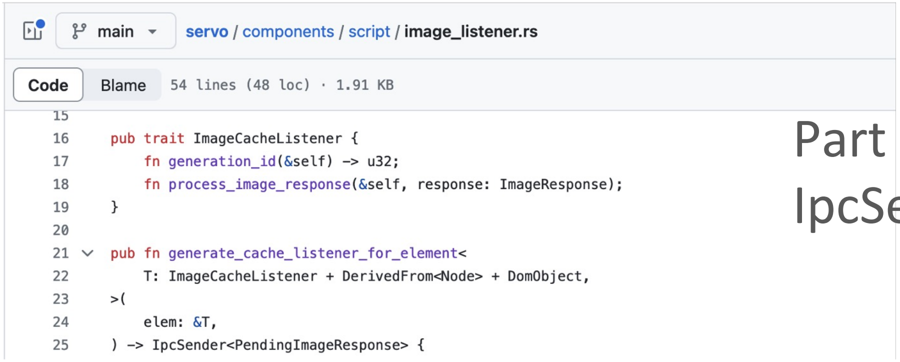
Code Blame 153 lines (132 loc) · 5.3 KB

```
pub trait ImageCache: Sync + Send {
    /// Add a listener for the provided pending image id, eventually called by
    /// ImageCacheStore::complete_load.
    /// If only metadata is available, Available(ImageOrMetadataAvailable) will
    /// be returned.
    /// If Available(ImageOrMetadataAvailable::Image) or LoadError is the final value,
    /// the provided listener will be dropped (consumed & not added to PendingLoad).
    fn track_image(
        &self,
        url: ServoUrl,
        origin: ImmutableOrigin,
        cors_setting: Option<CorsSettings>,
        sender: IpcSender<PendingImageResponse>,
        use_placeholder: UsePlaceholder,
    ) -> ImageCacheResult;

```

**Modularity**: 1: internal complexity hidden from other components?
Yes: only interfaces is shared(bonus: faster compilation).

## **Modularity**: 2: Can components be separated and recombined? No: listener concept(hidden in script component) introduces IPC dependency.


main

servo / components / script / image_listener.rs

Code Blame 54 lines (48 loc) · 1.91 KB

```rust

pub trait ImageCacheListener {
    fn generation_id(&self) -> u32;
    fn process_image_response(&self, response: ImageResponse);
}

pub fn generate_cache_listener_for_element<
    T: ImageCacheListener + DerivedFrom<Node> + DomObject,
>(
    elem: &T,
) -> IpcSender<PendingImageResponse> {
```

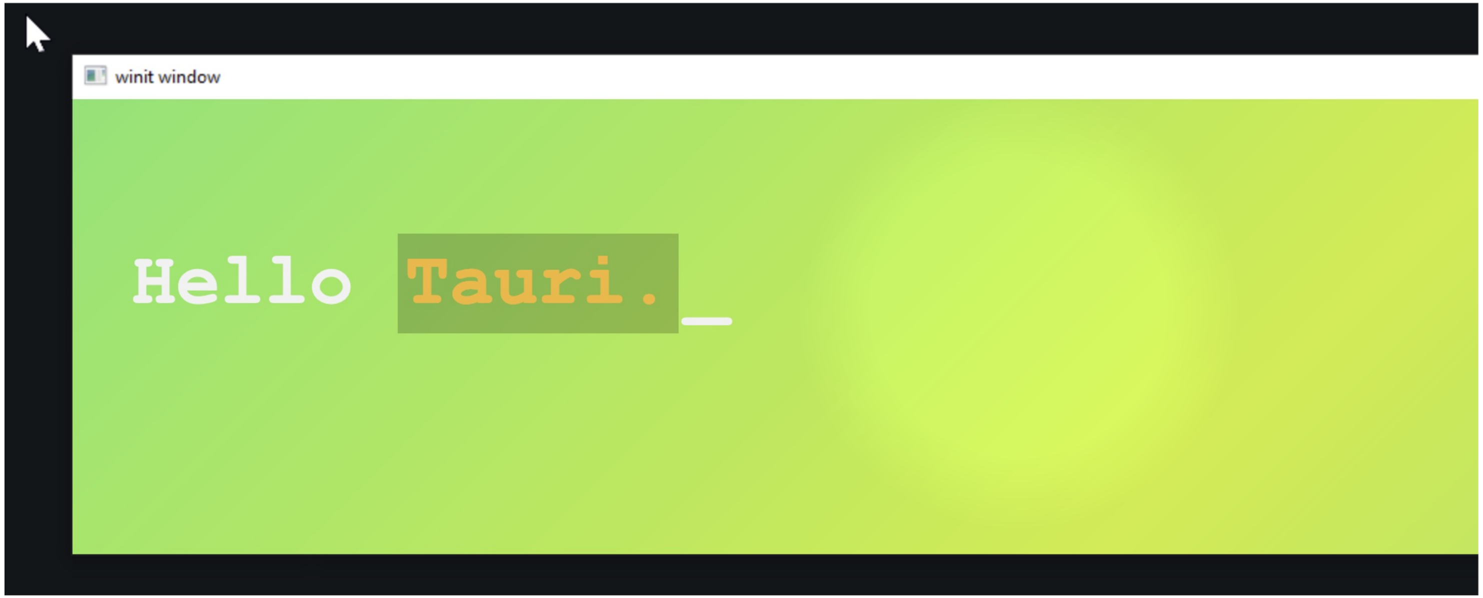
Part of the interface: IpcSender<PendingImageResponse>

But almost: just need to abstract away means of communication.

As a modular embedded component:

1. Do components hide complexity from each other? Yes: embedding API
2. Can components be separated and recombined? Yes: see various embedding examples

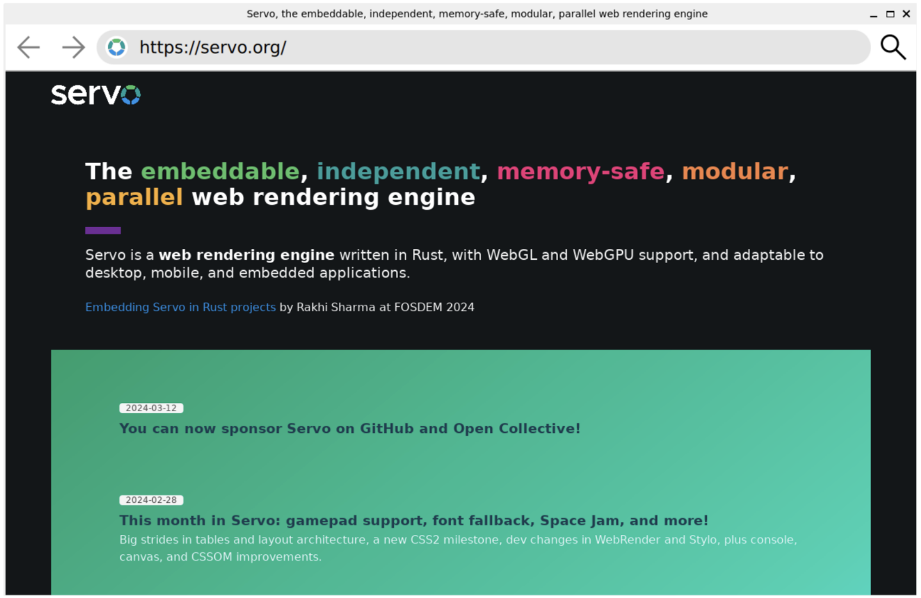
Examples: ServoShell, Tauri WebView, KDAB Qt WebView



Tauri WebView: project funded by NLnet.

*"The web ecosystem lacks a cross-platform, non-corporate controlled system for running web content. Tauri is a system for distributing cross-platform applications that relies on engines present on a system - effectively those owned by Apple, Google, and Microsoft. These permit varying levels of user control. The Servo project is a cross-platform, open source web engine."*

Source: https://nlnet.nl/project/Tauri-Servo/


Servo, the embeddable, independent, memory-safe, modular, parallel web rendering engine

https://servo.org/

servo

The embeddable, independent, memory-safe, modular, parallel web rendering engine

Servo is a **web rendering engine** written in Rust, with WebGL and WebGPU support, and adaptable to desktop, mobile, and embedded applications.

Embedding Servo in Rust projects by Rakhi Sharma at FOSDEM 2024

2024-03-12
**You can now sponsor Servo on GitHub and Open Collective!**

2024-02-28
**This month in Servo: gamepad support, font fallback, Space Jam, and more!**
Big strides in tables and layout architecture, a new CSS2 milestone, dev changes in WebRender and Stylo, plus console, canvas, and CSSOM improvements.

GOSIM 2024 EUROPE

## Qt WebView: project from KDAB

*“With the browser inherently being exposed to the internet, it is usually the biggest attack vector on a system. Naturally this makes Servo very attractive as an alternative browser engine, given that it is written in a memory-safe language.”*

“”At KDAB we managed to embed the Servo web engine inside Qt, by using our CXX-Qt library as a bridge between Rust and C++. This means that we can now use Servo as an alternative to Chromium for webviews in Qt applications.”

*Source:* https://www.kdab.com/embedding-servo-in-qt/

## Use of independent components:

1. Do components hide complexity from each other?
2. Can components be separated and recombined?

On both points: still a work in progress....

Examples: Spidermonkey, Webrender, WGPU

## Example: Spidermonkey

Script execution engine: JS and Wasm.

Some complexity hidden through safe Rust interfaces, but much use of low-level unsafe bindings still present.

Recent blog:

https://servo.org/blog/2024/04/15/spidermonkey/

Report: https://github.com/servo/servo/wiki/Servo-and-SpiderMonkey-Report

## Example: WGPU

Cross-platform, safe, pure-rust graphics api.

Used to implement WebGPU: DOM objects implemented by Servo in the script component, with a “backend” service running wgpu-core. Modular, again with the exception of some leaking: IPC communication.

Plans to re-use this infra to implement 2d canvas through Vello(Rust renderer using wgpu).

Example: Stylo-Blitz from Dioxus Labs

HTML and CSS renderer using Servo components: stylo(CSS resolution), html(html5ever) and css parsers(rust-cssparser).

“fulfill the long-held dream of many Rustaceans that Servo could power a native GUI library for Rust”,

https://github.com/jkelleyrtp/stylo-dioxus

## Example: Rust-url

URL parser for Rust.

Used both in Gecko networking stack(Necko), Servo, and about 300k cargo installs a month.

https://github.com/servo/rust-url

Servo, the embeddable, independent, memory-safe, **modular(?)**, parallel web rendering engine

Sometimes, and with ongoing efforts…

# THANK YOU

## More information available at:

@gterzian

@servo

GOSIM 2024 EUROPE

servo.org

# THANK YOU

GOSIM 2024 EUROPE